Efficient Approach for Query Optimization in Rough Data

Shruthi Hiremath[1], Pallavi Chandra[2]

[1, 2]School of Computer Science and Engineering, VIT University, Vellore-632014, Tamil Nadu, India

Abstract: ***In this paper we represent an efficient query optimization technique for the multi-valued rough relational database which follows the indiscernibility relation in its domain. This notion is perceived by using an encoding function to convert a multi-valued attribute to a constant single valued attribute. A simple select-querying technique is provided for selecting the tuples of single-valued attribute from a rough database .We extend the concept of query search to multi-valued attribute. .Here we use an encoding function to convert the multi-valued attribute to a single-valued constant attribute to optimize the query search and hence to reduce the response time.***

Keywords: rough set theory, rough relational database, indiscernibility relation, encoding function**.**

1. Introduction

The various applications that are run in the Software and IT industry are mainly uncertain .As a matter of fact any industry that functions today has a lot of uncertain and imprecise data which needs to be compiled. Rough Set Theory [2] was initially introduced by Pawlak to manage uncertain data and analyze the incomplete information effectively. A rough relational database [2] was later implemented to this effect. The major difference between a relational database and an RRDB is that the RRDB can consist of attributes comprising of one or more atomic values unlike relational database where only atomic values of attributes are dealt with. For querying data in a RRDB based on rough set theory it has been explained that for efficient query ,RRDB should be decomposed into standard relational table(semantics of query data is followed) and then use SQL and rough relational operators to get the results. This increases the time and space complexity Further a new concept was introduced where results were based on comparison between equivalence classes rather than values. In this paper we deal with an encoding function [3] which efficiently queries a data by converting a multi-valued attribute into a single-valued attribute thereby reducing the response time for both the lower approximation [2] (certain data) and upper approximation [2] (possible data).

2. Basic Concepts

2.1 Rough Set Theory

Let U be a non-empty set containing the set of all tuples called universal set and R defines an equivalence relation on the universal set U also called the indiscernibility relation. For defining a rough set we define a lower approximation and an upper approximation on a set X where $X \subseteq U$. Consider an ordered pair attribute A=(U,R).

Lower Approximation:

$\underline{R}X=\{x \in U \mid [x]_R \subseteq U\}$

This yields certain data.

Upper Approximation:

$\overline{R}X=\{x \in U \mid [x]_R \cap X \neq \varnothing \}$

This yields possible data.

2.2 Rough Relational Database

The rough relational database is similar to the relational database in terms that both comprise of data as a collection of relations containing tuples. These relations are also known as sets and are unordered and non-duplicated.

A relational database is defined as follows S = (U, A, D, R)

For any database U is a set of all tuples, representing the universal set.

A is the attribute set and D is the domain set.

2.2.1 Analysis for relational and rough database

In a classical relational database R is a relation defined over n sets $D_1, D_2, \dots, Dn$ where D_i represents a domain.

In a rough relational database R is equivalence classes defined on domain D. Consider $a_i \in A$, where d_{ai} is a domain defined on a_i, r_{ai} is an equivalence class on attribute a_i. The necessary condition for accessing a tuple t is that $t \in U$ where t (a_i) should access the value of the attribute a_i of that tuple t. This tuple will also be a $\subseteq D_{ai}$.

To define equivalence classes we make use of the indiscernibility relation.

From the table 1 given below the table consists of two attributes 'Manufacturers' and 'Products'.

Using the indiscernibility relation the equivalence classes can be defined as:

$R_{manufacturer}$ = [{P&G, Proctor & Gamble}, {Hindustan Unilever Limited, HUL}, {Britannia, Britannia Industries, Britannia Industries Limited}, {Nestle, Nestle S.A.}, {ITC, Indian Tobacco Limited, ITC Limited}]

$R_{product}$ = [{food, beverages}, {biscuits, confectionery, bakery products}, {dairy products}, {cleaning agents}, {personal care}, {tobacco}]

It is trivial from the 'manufacturer' relation that P&G and Proctor and Gamble represent the same company and hence are grouped under the same class.

Definition 1: A rough relation is a subset of the set cross product P (D_1) x P (D_2) x…P (D_n).
Definition 2: An interpretation α = (a_1, a_2,…,a_n) of a rough tuple t_i =(d_{i1},d_{i2},….,d_{in}) is any value assignment such that $a_j \in d_{ij}$ for all $1 \leq j \leq n$, a_j is called a sub-interpretation of d_{ij}.

In Table 1 we present a model of RRDB called 'FMCG'. Its two attributes are Manufacturers and Products.

Table 1: Fast Moving Consumer Goods (FMGC) Products

ID	*MANUFACTURERS*	*MANUFACTURER_BIN*	*PRODUCTS*	*PRODUCTS_BIN*
P001	P&G	10000	{Food, Beverages}	100000
P002	Proctor & Gamble	10000	{cleaning agents, personal care}	000110
H001	{P&G, Hindustan Unilever Limited}	11000	{Food , Beverages, cleaning agents, personal care }	100110
H002	HUL	01000	{beverages, food}	100000
H003	Hindustan Unilever Limited	01000	{cleaning agents, personal care}	000110
B001	Britannia	00100	{dairy products}	001000
B002	Britannia Industries	00100	{bakery products}	010000
B003	Britannia Industries Limited	00100	{biscuits }	010000
B004	{Britannia, Nestle}	00110	{dairy products, biscuits}	011000
N001	Nestle	00010	{dairy products}	001000
N002	Nestle S.A.	00010	{bakery products, dairy products}	011000
I001	ITC	00001	{Tobacco }	000001
I002	ITC Limited	00001	{Foods, confectionery}	110000
I003	Indian Tobacco Limited	00001	{personal care}	000010
I004	{ITC, HUL}	01001	{Food, personal care}	100010

Here, in this real-time application we have taken two attributes MANUFACTURER and PRODUCTS. After applying encoding function the attributes are extended to two more attributes namely MANUFACTURER_BIN and PRODUCT_BIN. The domain D and relation R for the table can be defined as:

$D_{manufacturer}$ = [P&G, Proctor & Gamble, Hindustan Unilever Limited, HUL, Nestle, Nestle S.A., ITC, Indian Tobacco Limited, ITC Limited, Britannia, Britannia Industries, Britannia Industries Limited]

$R_{manufacturer}$ = [{P & G, Proctor & Gamble}, {Hindustan Unilever Limited, HUL}, {Britannia, Britannia Industries, Britannia Industries Limited}, {Nestle, Nestle S.A.}, {ITC, Indian Tobacco Limited, ITC Limited}]

Dproducts = [Food, Beverages, Cleaning agents, Personal care, Dairy products, Bakery Products, Biscuits, Tobacco, Confectionery]

Rproducts = [{Food, Beverages}, {Biscuits, Confectionery, Bakery products}, {Dairy products}, {Cleaning agent}, {Personal care}, {Tobacco}]

3. The Encoding Function

To reduce the query response time we hereby define an encoding function for optimization:

1. Calculate the number of equivalent classes defined for each attribute a_i defined over a domain d (a_i).
2. Assign that many number of bits as calculated in (1).
e.g. Considering Rmanufacturer defined above the total number of equivalent classes is 5 and hence number of bits required for the encoding function will be 5.Initially all bits will be 0 i.e.00000.
3. For any given value of a tuple t_i for an attribute a_i, check in which equivalent class the value is present. Assign 1 to the position corresponding to the class if present else 0.
e.g. Consider Britannia with ID B001 is present in the 3rd equivalent class so its encoding function will have a bit 1 at the 3rd position i.e.00100.
4. Repeat the above steps and compute the encoded values for all the attributes in D_i.
5. If a multi-valued attribute is present then the encoding function for the same is defined using values of different equivalent classes defined on domain D_i .It will be computed by the OR operation of the individual encoded values calculated using (3).
For e.g. in table 1, consider {Britannia, Nestle} with ID B004, the encoding function will be OR of encoding values of Britannia and Nestle. Britannia has encoding value calculated as 00100 and Nestle has encoding value as 00010. Then encoded value of {Britannia, Nestle} will be 00110.

4. Algorithm

4.1. Algorithm 1

Suppose the origin select-condition is "a = v", a is an attribute and its domain is D_a, and a_BIN is the encoding filed of a; v is an arbitrary value and $v \subseteq D_a$.

1. Calculate the value of ENCODE (a, v) and note the result as c, that is c = ENCODE (a, v)

2. The search condition of certain data querying can be modified to “a_BIN = c”
3. The search condition of possible data querying can be modified to “a_BIN >= c ∧ a_BIN & c = c”, “a_BIN >= c” is an additional condition, and it can narrow the scope of search. According to the Algorithm 1, we can get the more common expression of our new method.

4.2. Algorithm 2

Suppose the origin select-condition is “$a_1 = v_1 \wedge a_2 = v_2 \ldots \wedge a_n = v_n$”, for all $1 \leq i \leq n$, a_i is an attribute and its domain is D_a, and a_i_BIN is the encoding filed of a_i; v_i is an arbitrary value and $v_i \subseteq D_a$.

1. For all $1 \leq i \leq n$, calculate the value of ENCODE (a_i, v_i) and note the result as c_i.

2. The search condition of certain data querying can be modified to “a_1_BIN = c_1 ∧ a_2_BIN = c_2 …∧ a_n_BIN = c_n”

3. The search condition of possible data querying can be modified to “(a_1_BIN >= c_1 ∧ a_1_BIN & c_1 = c_1) ∧ (a_2_BIN >= c_2 ∧ a_2_BIN & c_2 = c_2) … ∧ (a_n_BIN >= c_n ∧ a_n_BIN & c_n = c_n)”.

5. Experiment

To get the results for certain data we use the query format a_BIN = c.This is the lower approximation. Refer Figure 1.

To get the results of for possible data we use the query format a_BIN >= c ∧ a_BIN & c = c. This is the upper approximation. Refer Figure 2.

Consider the query: “select ID from FMCG where MANUFACTURER_BIN>=10000 and MANUFACTURER_BIN & 10000 = 10000”.

The result obtained after executing MANUFACTURER_BIN>=10000 is ID = {P001, P002, H001}. Now ANDing these values with 10000 we get possible data i.e. ID = {P001, P002, H001}.

Table 2: Certain Data Querying

Certain Data Querying	*Total Response Time*	
	Encode Algorithm	*Rough Algorithm*
1	2	3
2	4	6
3	6	9

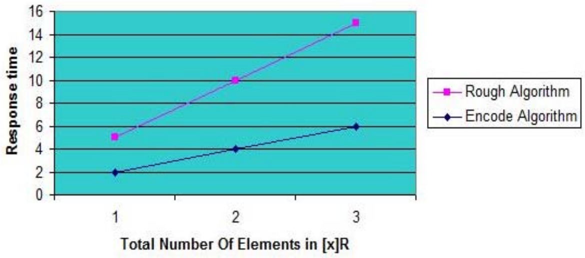

Figure 1: Certain Data Querying

Table 3: Possible Data Querying

Possible Data Querying	*Total Response Time*	
	Encode Algorithm	*Rough Algorithm*
1	2	3.67
2	4	5.67
3	6	9.50
4	8	

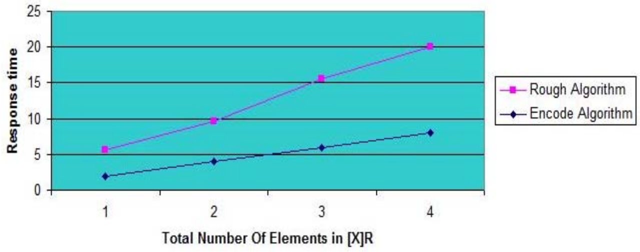

Figure 2: Possible Data Querying

6. Conclusion

In this paper we present a solution to reduce response time for querying multi-valued attribute in rough relational database. We have presented this by encoding multi-valued into single-valued attribute. We have also defined as sample query which makes use of the encoded values obtained after applying the encoding function on the attributes in the table.

7. Future Work

Further research could be carried out when an issue such as a conflict that arises when a query tries to access a tuple of any attribute a_i defined on a domain d_i which contains elements belonging to the same equivalent class, the problem that arises when we try to access is that it will not be able to distinguish which element it should retrieve. This problem may be solved by assigning separate bits for the equivalence classes.

References

[1] Pawlak Z. ”Rough Sets”. International Journal of Computer and Information science, 1982, 11(5): 341-356.
[2] PawlakZ. “Rough sets - theoretical aspects of reasoning about data”. Dordrecht: Kluwer Academic Publishers, 1991, pp. 68-162.
[3] T. Beaubouef, “Uncertainty processing in a relational database model via a rough set representation”. University Microfilms International, A Bell&Howell Information Company, PhD. dissertation, 1994, pp. 67-76.

[4] T. Beaubouef, Petry F, Aroar G. "Information theoretic measures of uncertainty for rough sets and rough relational databases". Information Science, 1998, 109:185-195.
[5] T. Beaubouef, F. Petry, and B. Buckles. "Extension of the relational database and its algebra with rough set techniques". Computational Intelligence, 1995, 11(2):233-245.
[6] Nakata M, Murai T. "Data Dependencies over Rough Relational Expressions". In: IEEE Intl. Fuzzy Systems Conf, 2001, pp. 1543-1546.
[7] Qiusheng An, Guoyin Wang, Junyi Shen, Jiusheng Xu. Querying Data from RRDB Based on Rough Sets Theory. LNAI2639, Springer-Verlag, 2003, pp. 342-345.
[8] Fuyuan Cao, Jiye Liang. "The Rough Data Query Based on SQL Language", Computer Science, 2004, VOL.31 No.2.
[9] Qiusheng An, Yusheng Zhang, Wenxiu Zhang. "The study of rough relational database based on granular computing". Granular Computing, 2005 IEEE International Conference on Granular Computing, July 2005, VOL. 1: 108~111.

Author Profile

Shruthi Hiremath is a third year undergrad student currently pursuing her Bachelor's degree in Computer Science and Engineering from Vellore Institute of Technology, Vellore She has an aptitude for research work and is currently working on a project concerning provision of health-care solutions to the employees of the municipal corporation of the Pune city, Maharashtra, India. She aspires to do a Master's degree in Computer Engineering, her subjects of interest being Database Systems, Rough Sets, Software Engineering, Soft Computing and Computer Networks.

Pallavi Chandra is a third year undergrad student also currently pursuing her Bachelor's degree in Computer Science and Engineering from Vellore Institute of Technology, Vellore. Her areas of Interest lie in Cryptography, Database Systems and Rough Sets. She is presently working on a project with BSNL. She wishes to appear for the Indian Administration Examinations.